UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

ADQUISICIÓN DE DATOS CON MICROCONTROLADORES PARA APLICACIONES EN EL INTERNET DE LAS COSAS

AUTOR: Blanca de Angulo Ruiz-Morón
TUTOR: Guillermo Robles Muñoz
Junio de 2017

INDICE DE CONTENIDOS

INDICE DE ILUSTRACIONES ..... - 4 -
INDICE DE TABLAS ..... - 4 -
1. INTRODUCCIÓN ..... - 5 -
1.1. OBJETIVOS ..... - 6 -
2. HARDWARE ..... - 7 -
2.1. PLACA DE DESARROLLO ..... - 7 -
2.1.1. MICROCONTROLADOR STM32L4 ..... - 7 -
2.1.2. PLACA DE DESARROLLO. NUCLEO-L432KC ..... - 13 -
2.1.3. COMPARATIVA DEL NÚCLEO-L432KC CON ARDUINO ..... - 15 -
2.2. PERIFÉRICOS ..... - 15 -
2.2.1. SENSOR BMP280 ..... - 15 -
2.2.2. PANTALLA OLED CON MICRO SSD1306 ..... - 17 -
3. CONEXIÓN ENTRE LA PLACA DE DESARROLLO Y LOS PERIFÉRICOS ..... - 18 -
3.1. DISEÑO Y CONFIGURACIÓN DEL HARDWARE ..... - 18 -
3.1.1. NÚCLEO-L432KC ..... - 18 -
3.2. CONEXIONADO ..... - 20 -
3.2.1. NÚCLEO CON SENSOR ..... - 20 -
3.2.2. NÚCLEO CON PANTALLA ..... - 21 -
3.2.3. PULSADOR ..... - 22 -
3.2.3.1. NÚCLEO CON PULSADOR ..... - 24 -
3.2.4. CONEXIONADO DEL CONJUNTO ..... - 24 -
4. SOFTWARE ..... - 26 -
4.1. HERRAMIENTAS DE DESARROLLO DE SOFTWARE ..... - 26 -
4.1.1. IDEs ..... - 26 -
4.1.2. ST-LINK / V2-1 ..... - 27 -
4.1.3. RTC ..... - 28 -
4.2. PROGRAMA PRINCIPAL ..... - 28 -
4.2.1. LIBRERÍAS PARA LOS PERIFÉRICOS ..... - 29 -

4.2.2. FUNCIONES ..... - 31 -

4.2.3. PROGRAMA PRINCIPAL ..... - 35 -

4.3. DIAGRAMA DE FLUJO ..... - 37 -

5. PRESUPUESTO ..... - 39 -

6. CONCLUSIONES Y TRABAJOS FUTUROS ..... - 40 -

7. BIBLIOGRAFÍA ..... - 41 -

8. ANEXOS ..... - 43 -

8.1. CÓDIGO ..... - 43 -

INDICE DE ILUSTRACIONES

Ilustración 1. Familia STM32 ..... - 8 -
Ilustración 2. Tipo de microcontroladores STM32L4 ..... - 9 -
Ilustración 3. Bus SPI ..... - 10 -
Ilustración 4. Funcionamiento Bus I2C. ..... - 11 -
Ilustración 5. Diagrama de la entrada al convertidor ADC. ..... - 12 -
Ilustración 6. Diagrama de bloques del Núcleo-L432. ..... - 13 -
Ilustración 7. Conector CN1 para Núcleo-L432. ..... - 19 -
Ilustración 8. Pines Sensor BMP280 ..... - 20 -
Ilustración 9. Pines de la Pantalla Oled ..... - 21 -
Ilustración 10. Funcionamiento del pulsador ..... - 22 -
Ilustración 11. Pulsador ..... - 22 -
Ilustración 12. Circuito Pull-Up con botón pulsado ..... - 23 -
Ilustración 13. Circuito Pull-Up con botón sin pulsar ..... - 23 -
Ilustración 14. Circuito NRST del Núcleo-L432 con protección de clavija NRST recomendada. ..... - 24 -
Ilustración 15. Conexionado del conjunto con Fritzing ..... - 25 -
Ilustración 16. Fotografía del conjunto ..... - 25 -
Ilustración 17. Diagrama de Flujo General ..... - 38 -

INDICE DE TABLAS

Tabla 1 ..... - 18 -
Tabla 2 ..... - 19 -
Tabla 3 ..... - 20 -
Tabla 4 ..... - 21 -
Tabla 5 ..... - 24 -
Tabla 6 ..... - 39 -

1. INTRODUCCIÓN

El internet de las cosas es un concepto, cada vez más popular en los últimos tiempos, referido a la interconexión de varios dispositivos a través de Internet. Por ejemplo, objetos de la vida cotidiana conectados entre sí y a Internet que ofrecen datos en tiempo real. Se crean dispositivos con sensores para registrar datos, analizarlos y procesarlos por un microcontrolador, y posteriormente enviar esos datos a la red.

La expresión Internet of Things (IoT), fue designada por Kevin Ashton, profesor del MIT, en el año 2009. Este concepto ha estado muy presente en los últimos años, pero sus orígenes datan del año 1926, cuando Nikola Tesla conformó las bases de las comunicaciones inalámbricas y de la radio. Fue en 1969, cuando ARPANET, la primera red de conexión informática, envió el primer mensaje entre las universidades de Standford y Ucla. [1]
En los siguientes años, se siguió evolucionando, pero la revolución llegó en el siglo XXI con la entrada de la conectividad inalámbrica. Esto permitió un crecimiento en los objetos conectados, de los que se han derivado nuevos conceptos como el M2M (Machine to Machine), para concluir con IoT. [2]
Existen numerosas aplicaciones de IoT para los diferentes sectores como el sector sanitario, el comercio, la agricultura, la ganadería, etc… La tecnología de IoT también es aplicable en los hogares, y producirá un cambio en nuestro estilo de vida. Las fábricas mejoraran sus procesos productivos. Gracias a la conectividad, la disponibilidad de información, se aumenta la innovación y se aumentará la productividad en cualquier sector.
En cuanto al impacto económico, según Cisco System en el año 2020 habrá cerca de 50.000 millones de dispositivos conectados a Internet generando un negocio de 10 billones de dólares. [3]

Los avances de IoT implican nuevas tecnologías para el sistema de gestión de datos, como sistemas de Big Data. Hoy en día, empresas trabajan en la creación de protocolos de comunicación, pues es fundamental que el protocolo de comunicación entre los dispositivos sea seguro para la protección de la integridad de los datos.
En la evolución del Internet de las Cosas, es indispensable el bajo consumo del sensor y del microcontrolador para la autonomía de los dispositivos.

En este proyecto, se ha desarrollado un sistema más enfocado a dispositivos de actividad. Concretamente, se ha utilizado un sensor para la adquisición de datos de temperatura y presión atmosférica. A través de un microcontrolador, conectado al sensor, se procesa la información que será mostrada por un dispositivo de salida, una pantalla.

Se ha decidido trabajar con un microcontrolador de la serie STM32L4 por sus numerosas ventajas, como sus altos rendimientos. El microcontrolador está integrado en el Núcleo-L432kc que presenta periféricos inteligentes.

Los periféricos con los que se ha desarrollado el proyecto son: el sensor BMP280 y la pantalla OLED (Organic light-emitting diode) con micro SSD1306. Ambos presentan bajos consumos de energía, que los hacen interesantes en aplicaciones que necesitan de baterías para su funcionamiento.

El software, mostrado por la pantalla OLED, se ha desarrollado e implementado en la plataforma "mbed". Esta plataforma es muy flexible y facilita la creación de prototipos trabajando con el microcontrolador STM32L432KC.

El diseño del software se basa en un menú que, con la ayuda de un interruptor, el usuario se desplaza por él, mostrando las distintas opciones. En otro apartado se explicará con detalle el diseño del menú.

1.1. OBJETIVOS

- Como principal objetivo se ha establecido el desarrollo de un sistema autónomo para la adquisición de datos de temperatura y presión a partir de un sensor, y mostrarlos por una pantalla.
- Diseñar un software, basado en un menú que se muestre por pantalla, y usar un pulsador para moverse por el menú.
- Búsqueda de un periférico con protocolo estándar SPI para conectarlo con la placa de desarrollo.
- Conseguir que el sistema tenga un consumo de energía muy reducido, alimentándose con tensiones bajas.
- Bajo coste. Trabajar con dispositivos baratos y fáciles de adquirir.

2. HARDWARE

2.1. PLACA DE DESARROLLO

2.1.1. MICROCONTROLADOR STM32L4

Los microcontroladores STM32L4 son los microcontroladores de ultra baja potencia basados en el núcleo RISC ARM Cortex-M4 que trabajan con un alto rendimiento y operando a una frecuencia de hasta 80 MHz. Este microcontrolador es muy eficaz por la baja potencia con la que se alimenta.

La arquitectura RISC (Reduced Instruction Set Computer) ejecuta las órdenes simples de forma rápida y eficientemente de forma que se consigue una mayor velocidad de procesamiento.

ARM (Advanced Risc Machine) es una arquitectura RISC de 32 bits y recientemente de 64 bits desarrollada por la empresa ARM Holdings.

El núcleo Cortex-M4 es la arquitectura más reciente para sistemas integrados, cuenta con una unidad de punto flotante (FPU) y con capacidades para el procesamiento de señal digital (DSP). Fue desarrollado para proporcionar una plataforma de bajo coste que satisfaga las necesidades de la implementación de la MCU. [7].

La familia STM32 de microcontroladores de memoria Flash de 32 bits está basada en el procesador ARM Cortex-M y están disponibles en diferentes líneas: STM32L4x1 (línea de acceso), STM32L4x2 (dispositivo USB), STM32L4x3 (dispositivo USB, LCD), STM32L4x5 (USB OTG) y STM32L4x6 (USB OTG, LCD). Se diferencia por el núcleo Cortex-M que utilicen y el consumo de potencia.

En la Ilustración 1, se muestra la familia STM32. En ella se puede ver que el microcontrolador STM32L4 es uno de los que presentan un consumo de energía ultrabajo (ultra-low-power) y además es una de las versiones del microcontrolador Cortex-M4.

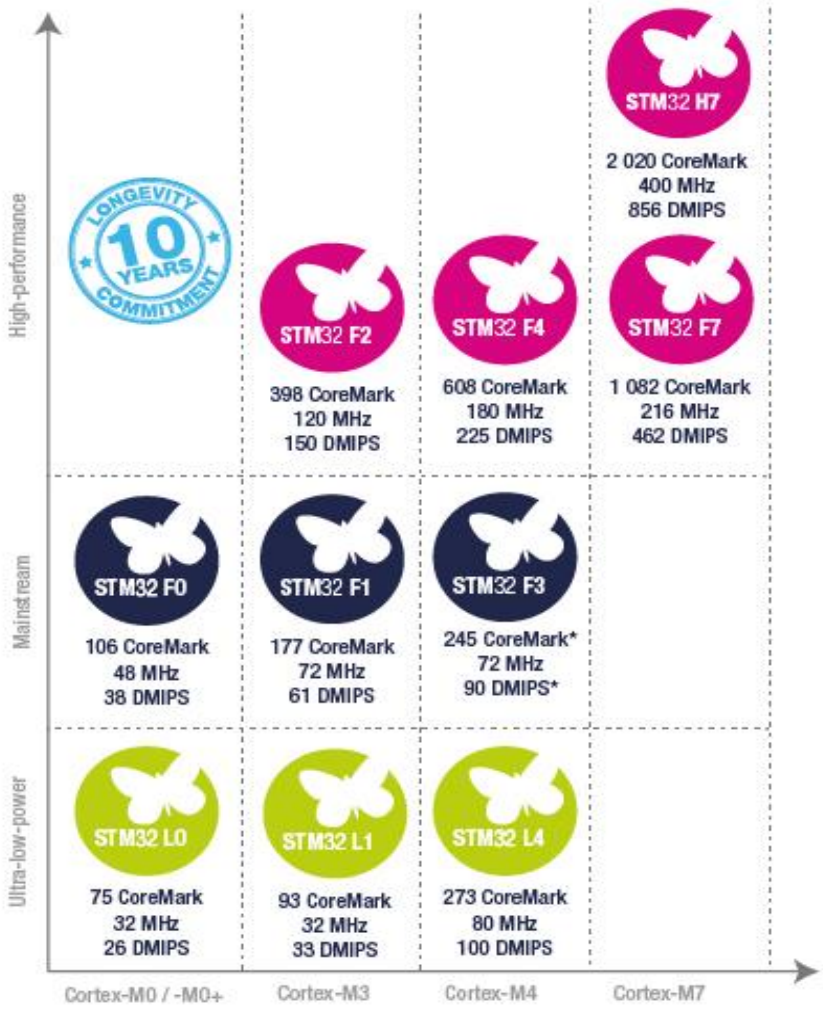

Ilustración 1. Familia STM32

La frecuencia de funcionamiento, es decir, la velocidad del procesador es de 80 MHz como máximo, pudiéndose disminuir por software.

La tensión de alimentación varía entre 1.65 V y 3.6 V. El dispositivo permanece en modo de reposición cuando la tensión de alimentación monitorizada VDD está por debajo del umbral especificado, sin necesidad de un circuito de reposición externo. Estos dispositivos ofrecen un escalado de voltaje dinámico para equilibrar el consumo de energía con la demanda de procesamiento y periféricos de baja potencia. [4]

El objetivo es alimentar el microcontrolador con tensiones bajas que permite reducir su consumo e incorporar fuentes alternativas como sistemas de captura de energía como el "Energy Harvesting".

Estos microcontroladores cuentan con 256 Kbytes de memoria Flash y con 64 Kbyte de SRAM

incorporada. La memoria Flash de tipo no volátil, permite almacenar grandes cantidades de memoria en un espacio reducido a bajo coste. La memoria SRAM (Static Random Access Memory) es de tipo volátil, es decir, la información guardada en esta memoria se borra cuando se interrumpe la alimentación. Este tipo de memoria mantiene los datos sin necesidad de circuito de refresco, siempre que esté alimentada.

En la Ilustración 2, se puede ver la clasificación los distintos microcontroladores de tipo STM32L4 según la memoria Flash y RAM.

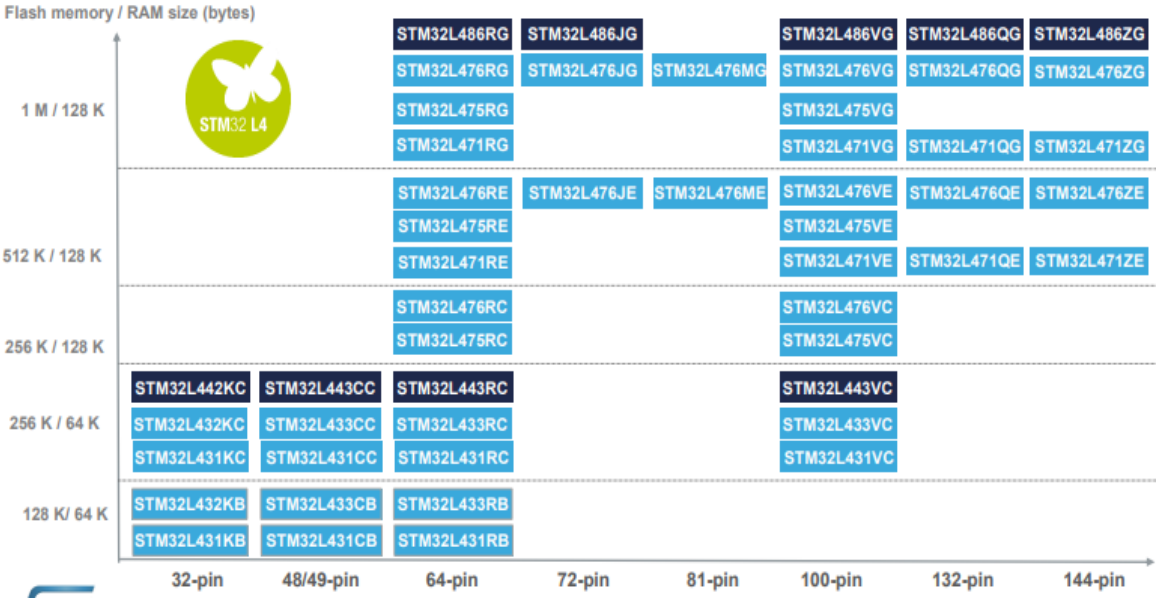

Ilustración 2. Tipo de microcontroladores STM32L4

Los dispositivos STM32L432xx incorporan varios mecanismos de protección para la memoria Flash incorporada y la SRAM: protección de lectura, protección contra escritura, protección de lectura de código propietario y cortafuegos. [5]

El dispositivo tiene integrado un RTC, un reloj de tiempo real de baja potencia. El RTC es un temporizador/contador independiente y recibe energía del suministro de VDD.

- Bus SPI

El microcontrolador se puede comunicar con periféricos por SPI, es posible conectar hasta dos periféricos. El bus SPI (Serial Peripheral Interface), desarrollado en 1980 por Motorola, es un protocolo síncrono que trabaja en modo "full dúplex", es decir, se recibe y se transmite la información. Es un bus síncrono porque todos los dispositivos están sincronizados, debido a que el dispositivo maestro proporciona una señal de reloj.

Su arquitectura es de tipo maestro-esclavo, es decir, el dispositivo maestro, envía y recibe datos de los dispositivos esclavos. Los dispositivos esclavos no pueden intercambiar directamente los datos entre ellos.

En este caso, el microcontrolador se comunica con la pantalla OLED a través del bus SPI. El dispositivo maestro es el microcontrolador y la pantalla es el esclavo. Estos se comunican entre sí al mismo tiempo utilizando canales o líneas diferentes en el mismo cable. El microcontrolador transmitirá la información a los esclavos, que recibirá y enviará la información al microcontrolador. De este modo se realiza una conversión paralela para transmitir información.

En la Ilustración 12, se representa el funcionamiento del bus SPI. El maestro mantiene el estado HIGH (1), y cambia a LOW (0) para mantener la comunicación con el esclavo. El maestro envía 1 bit al esclavo al mismo tiempo que recibe otro del esclavo, esto ocurre en cada pulso de la señal de reloj, en el flanco de subida. [6]

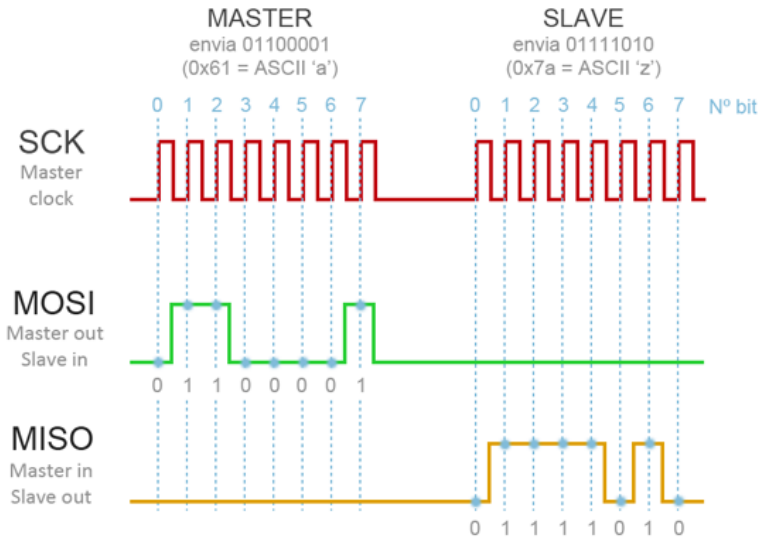

Ilustración 3. Bus SPI.

- Bus I2C

El bus I2C (Inter-Integrated Circuit) es un protocolo de comunicación serie diseñado en 1982 por Philips. Es síncrono, pues el maestro mantiene sincronizados a los dispositivos proporcionando una señal del reloj. Es más ventajosa que la UART pues no hace falta establecer mecanismos de transmisión para mantener la transmisión sincronizada.

Su arquitectura es de tipo maestro-esclavo. Al igual que el bus SPI, el maestro es el que inicia la comunicación y puede mandar o recibir datos de los esclavos. En este proyecto, el microcontrolador se comunica por el bus I2C con el sensor BMP280, de tal forma que la comunicación es enviada por el microcontrolador, y es el sensor BMP280 quien recibe la información.

Este tipo de comunicación es menos ventajosa pues no es full dúplex como el SPI y su velocidad de comunicación es baja.

En la ilustración 10 se muestra el funcionamiento del bus I2C. De forma que usa 1 bit de validación, 7 bits a la dirección del esclavo, 1 bit para indicar si enviar o recibir información y 1 o más bits que son los datos que el esclavo envía o recibe. [7]

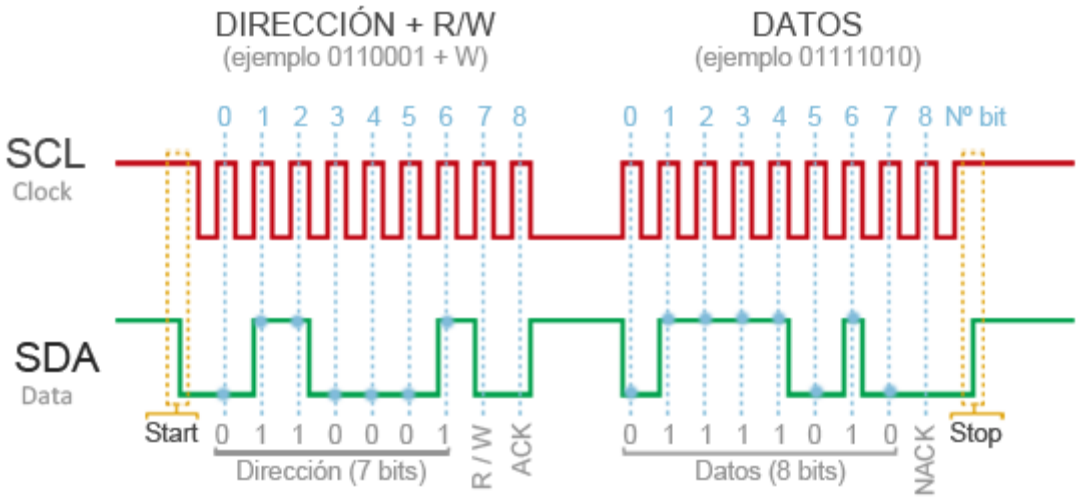

Ilustración 4. Funcionamiento Bus I2C.

- Convertidor de analógico a digital (ADC)

Este microcontrolador es capaz de convertir señales analógicas, como datos de temperatura captados por el sensor, en señales digitales tales como cambiar el estado de un LED a encendido (1). Puede trabajar hasta con 10 canales externos y permite obtener señales de muy buena resolución de 12 bits. En la Ilustración 2, se representa un diagrama de convertidor ADC.

En la Ilustración 3 se muestra un diagrama de la entrada al convertidor ADC. La parte izquierda del circuito consta de un filtro paso bajo, el cual deja pasar las bajas frecuencias atenuando a su vez las altas frecuencias. Si el valor del condensador $C_{parasitic,}$ es alto, se disminuye la exactitud de la conversión.

Pasando la entrada analógica, se encuentran dos diodos que permiten la circulación de la corriente en un sólo sentido y una fuente independiente de intensidad cuyo valor representa la pérdida de corriente de entrada de los pines IO.

Por último, dentro del convertidor ADC tenemos una resistencia y un condensador cuya frecuencia de reloj se reduce para mejorar la exactitud de la señal.

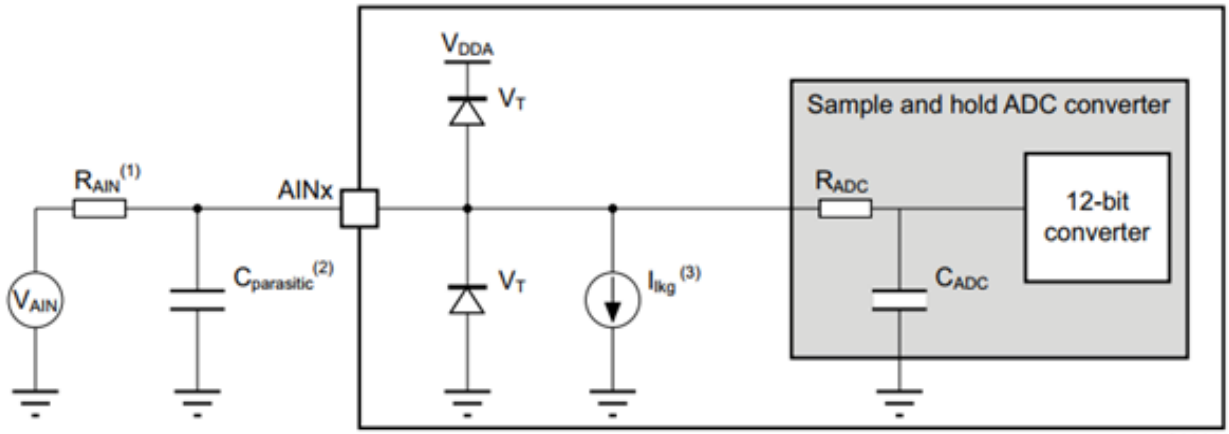

Ilustración 5. Diagrama de la entrada al convertidor ADC.

- GPIO con capacidad de interrupción externa

Consta de 20 pines GPIO (General Purpose Input Output), es decir, pines que pueden ser configurados por software como salida o entrada, o como función periférica alterna. La mayoría de los pines GPIO se comparten con funciones alternativas digitales o analógicas. Los valores de entrada que se pueden leer, estos son generalmente son HIGH (1) y LOW (0). [8]

2.1.2. PLACA DE DESARROLLO. NUCLEO-L432KC

El Núcleo-L432KC, basado en el microcontrolador STM32L4, ofrece de forma asequible y flexible la posibilidad de construir prototipos con cualquier línea de microcontroladores STM32. Este núcleo combina un alto rendimiento con un bajo consumo de energía.

Como se ha explicado en el apartado anterior, el microcontrolador STM32L4 es de la serie de los ultra-low-power, es decir, con un consumo de energía ultrabajo. En la Ilustración 4, muestra las conexiones entre el microcontrolador STM32 y sus periféricos (conectores ST-LINK / V2-1, pulsador, LED y Arduino Nano).

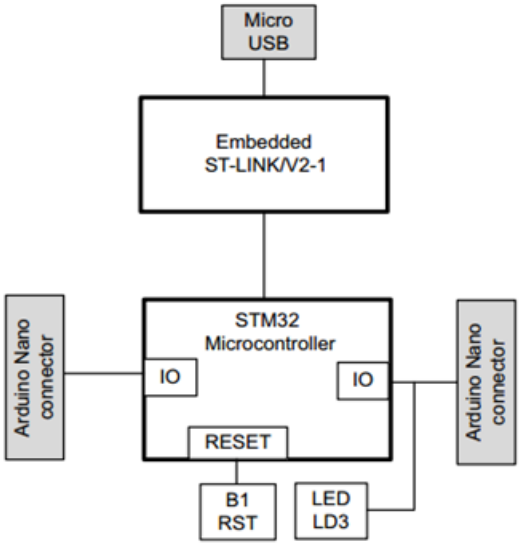

Ilustración 6. Diagrama de bloques del Núcleo-L432.

El núcleo cuenta con los siguientes accesorios:

- Los microcontroladores STM32 en paquetes de 32 pins.
- Extensión con conectividad nano de Arduino.
- mbed habilitado. La programación ha sido desarrollada en mbed.org.
- Depurador / programador integrado en la placa ST-LINK/V2-1 con conector SWD.
    - Interruptor de modo de selección para usar el kit como un ST-LINK / V2-1 independiente.
    - Tres interfaces diferentes soportadas en USB: puerto COM virtual, almacenamiento masivo, puerto de depuración.
- Fuente de alimentación flexible de la placa
    - VBUS USB o fuente externa (3,3 V, 5 V, 7 - 12 V).
    - Punto de acceso de gestión de energía.
- Tres LEDs
    - Comunicación USB (LD1), LED de alimentación (LD2), LED de usuario (LD3).
- Botón de Reinicio. La placa tiene un único pulsador conectado a NRST, y se utiliza para reiniciar el STM32. [9]

2.1.3. COMPARATIVA DEL NÚCLEO-L432KC CON ARDUINO

Arduino es una placa con un microcontrolador integrado, normalmente de la familia Atmel, con puertos digitales y analógicos de entrada y salida.

Existen diferentes versiones de Arduino, de las más recientes y novedosas es la placa Arduino ZERO. Se va a realizar de forma breve una comparación de esta placa con el Núcleo-L432KC para explicar la elección de la placa de desarrollo de este proyecto.

La placa Arduino ZERO, está alimentada por una MCU SAMD21 de Amtel, y a diferencia de las versiones anteriores, presenta un núcleo de 32 bits ARM Cortex M0+. Este núcleo, con el menor tamaño de todos, tiene unas características similares a las del Cortex M4, como su bajo consumo de energía.

A diferencia del Cortex M4, el Cortex M0+ no presenta una unidad de punto flotante (FPU) y capacidades para el procesamiento de señal digital (DSP).

El cortex M0+ es una arquitectura únicamente de 32 bits. En cambio, el Cortex M4 también adopta 64 bits.

Ambos procesadores tienen 256 Kb de memoria Flash, pero el Núcleo-L432kc presenta una memoria SRAM de 64 Kb y el Arduino ZERO solo de 32 kb.

La diferencia fundamental entre estas placas de desarrollo es la velocidad del reloj con la que puede operar es hasta 80 MHz. El microcontrolador del Arduino Zero trabaja con una frecuencia hasta 48 MHz. [10]

2.2. PERIFÉRICOS

2.2.1. SENSOR BMP280

BMP280 es un sensor ambiental, diseñado por Bosch, que mide la presión barométrica y la temperatura. Se basa en la tecnología de sensor de presión piezoresistiva, ofreciendo alta precisión, linealidad y flexibilidad para optimizar el consumo de energía, la resolución y el rendimiento del filtro. Se proporcionan un conjunto de ajustes predeterminados para facilitar el diseño de entrada.

Este sensor de Bosch es una solución de bajo coste para medir la presión barométrica con una precisión absoluta de ± 1 hPa y una temperatura con precisión de ± 1,0 ° C.

Se ha elegido este sensor por su bajo consumo de energía y sus pequeñas dimensiones que permiten la implementación en dispositivos que son alimentados por baterías. El módulo sensor se aloja en un paquete extremadamente compacto. [11]

El BMP280 consta de un elemento de sensor de presión y un ASIC (Circuito integrado de aplicación específica) de señal mixta que realiza conversiones de analógico a digital.

El sensor ofrece flexibilidad al diseñador y se puede adaptar a los requisitos de precisión, consumo de energía y tiempo de medición combinando los ajustes del sensor.

BMP280 opera en tres modos de potencia: modo de reposo, modo normal y modo forzado. En el modo de reposo, no se realizan mediciones. El modo normal comprende un ciclo perpetuo automatizado entre un periodo de medición activo y un período de espera inactivo. En el modo forzado, se realiza una sola medición. Una vez finalizada la medición, el sensor vuelve al modo de reposo.

Está disponible un conjunto de ajustes de sobremuestreo que van desde la configuración de ultra baja potencia hasta la resolución ultra alta para adaptar el sensor a la aplicación de destino. Los ajustes son combinaciones predefinidas de sobremuestreo de medición de presión y medición de temperatura.

A continuación, se detallan las especificaciones del sensor:

- El tipo de presión es absoluta y la presión de operación es 300 hPa - 1100 hPa
- Precisión: 1 hPa
- Tipo de salida: digital
- Temperatura operativa máxima: +85 ºC y mínima: -40 ºC.
- Tensión de alimentación VDDIO: 1.2 - 3.6 V
- Tensión de alimentación VDD: 1,71 - 3,6 V
- Resolución de datos de presión: 0.01 hPa y de temperatura: 0.01 ° C
- Voltaje operativo de suministro: 12 V
- Consumo de corriente media: 2,74 µA, típico (modo de alimentación ultrabaja)
- Consumo medio de corriente en modo de reposo 0,1 µA
- Tiempo promedio de medición: 5,5 mseg (preselección de potencia ultrabaja)
- Interfaz: I2C y SPI.

  En este proyecto, para comunicar el sensor con el núcleo utilizamos la interfaz I2C. El sensor actúa como esclavo y el núcleo como maestro. [12]

2.2.2. PANTALLA OLED CON MICRO SSD1306

La pantalla está hecha con 128x64 puntos LED (light-emiting diode) individuales, cada punto es encendido o apagado por su controlador interno. Al contrario que la pantalla LCD, las pantallas OLED son más competitivas por las características que ofrecen, como su alto contraste y alto brillo, auto-luminoso, un amplio ángulo de visión, amplio rango de temperatura de funcionamiento de -40ºC a 85ºC y su bajo consumo de energía.

El driver interno es un SSD1306 que se puede comunicar mediante I2C o SPI. Dado que para conectarlo por I2C es necesario realizar una soldadura, en este proyecto la pantalla se comunica por SPI con el microcontrolador STM32L4. Como he explicado anteriormente, la pantalla recibirá y enviará la información al microcontrolador.

El controlador SSD1306 se integra con control de contraste, RAM de visualización y oscilador, lo que reduce el número de componentes externos y el consumo de energía.
[13]

La fuente de alimentación:

- VDD = 1.65 V a 3.3 V (para la lógica IC, circuitos integrados).
- VCC = 7 V a 15 V (para la conducción de paneles).

Para visualización matricial

- Voltaje de salida de conducción OLED, máximo de 15V
- Corriente de fuente máxima del segmento: 100uA
- Máxima corriente de fregadero común: 15mA

Frecuencia de fotogramas programable y relación de multiplexación

Los drivers y las librerías que hay que instalar están disponibles en la página de “Adafruit”. Esto simplifica al usuario trabajar con la pantalla pues solo hay que adaptar el periférico.

La elección de esta pantalla mini se debe a su bajo coste y consumo de energía y a que es adecuada para aplicaciones portátiles compactas.
Además del bajo coste y bajo consumo de energía, se ha elegido esta pantalla porque es adecuada para aplicaciones portátiles compactas y porque permite la comunicación por SPI.

3. CONEXIÓN ENTRE LA PLACA DE DESARROLLO Y LOS PERIFÉRICOS

3.1. DISEÑO Y CONFIGURACIÓN DEL HARDWARE

3.1.1. NÚCLEO-L432KC

En la siguiente tabla se muestra los conectores de la placa de desarrollo

CONECTOR IZQUIERDO			
NÚMERO DE PIN	**NOMBRE DE PIN**	**STM32 PIN**	**FUNCIÓN**
1	D1	PA9	USART1_TX
2	D0	PA10	USART1_RX
3	RESET	NRST	RESET
4	GND	-	Ground
5	D2	PA12	-
6	D3	PB0	TIM1_CH2N
7	D4	PB7	
8	D5	PB6	TIM16_CH1N
9	D6	PB1	TIM1_CH3N
10	D7	PC14	
11	D8	PC15	
12	D9	PA8	TIM1_CH1
13	D10	PA11	SPI_CS || TIM1_CH4
14	D11	PB5	SPI1_MOSI || TIM
15	D12	PB4	SPI1_MISO
CONECTOR DERECHO			
NÚMERO DE PIN	**NOMBRE DE PIN**	**STM32 PIN**	**FUNCIÓN**
1	VIN	-	Power input
2	GND	-	Ground
3	RESET	NRST	RESET
4	+5V	-	5v input/output
5	A7	PA2	ADC12_IN7
6	A6	PA7	ADC12_IN12
7	A5	PA6	ADC12_IN11 || I2C1_SCL
8	A4	PA5	ADC12_IN10 || I2C1_SDA
9	A3	PA4	ADC12_IN9
10	A2	PA3	ADC12_IN8
11	A1	PA1	ADC12_IN6
12	A0	PA0	ADC12_IN5
13	AREF	-	AVDD
14	+3V3	-	3.3V input/output
15	D13	PB3	SPI1_SCK

Tabla 1 *[9]*

Para alimentar el núcleo, se conecta a un PC a través del conector USB CN1 con un cable USB. Para conectarlo a batería, se conecta al pin VIN.
Una vez conectado se encienden el LED1 y LED2 (LEDs rojos) y el LED3 parpadea (LED verde). El conector CN1 se indica dentro del círculo rojo en la ilustración 7.

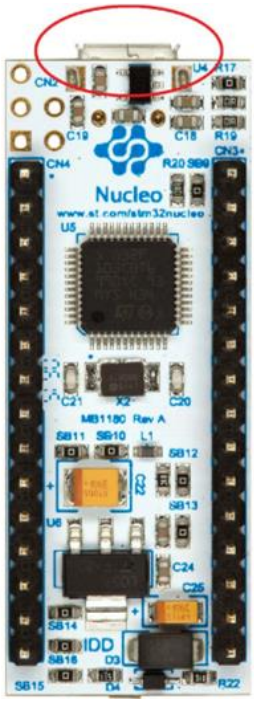

Ilustración 7. Conector CN1 para Núcleo-L432.

La placa STM32 y sus placas de protección pueden alimentarse de tres formas distintas de una fuente de alimentación externa, dependiendo de la tensión utilizada.

Nombre de la potencia de entrada	**Conector PIN**	**Rango de tensión**	**Corriente máxima**
VIN	CN4 pin 1	7 V - 12 V	800 mA
+5 V	CN4 pin 4	4.75 V – 5.25 V	500 mA
+3V3	CN4 pin 14	3 V – 3.6 V	-

Tabla 2

3.2. CONEXIONADO

3.2.1. NÚCLEO CON SENSOR

A continuación, se explica la conexión de los pines entre la placa de desarrollo y el sensor. [14] La Ilustración 9 muestra una imagen de los pines que tiene el sensor.

Ilustración 8. Pines Sensor BMP280

En la tabla 3 se muestra la conexión de los pines del sensor con la del núcleo. El sensor se alimenta con 3V3.

La comunicación con el microcontrolador se produce mediante la interfaz de bus I2C. Simplemente hacen falta dos conexiones, una con la señal de reloj (SCL) y otra para la transmisión de datos (SDA). El núcleo ejerce de maestro, quien controla el reloj, y el sensor de esclavo.

NÚCLEO-L432KC	**SENSOR BMP280**
+3V3	**VCC**
GND	**GND**
(A5) I2C1_SCL	**SCL**
(A4) I2C2_SDA	**SDA**

Tabla 3

3.2.2. NÚCLEO CON PANTALLA

Ilustración 9. Pines de la Pantalla Oled.

En este tipo de comunicación, el maestro genera el reloj y se encarga de enviar la información al esclavo. En este caso, el núcleo es el maestro y la pantalla el esclavo.

El envío de información se hace por una línea de datos llamada MOSI (Master out slave in) y a través de la línea llamada MISO (Master in slave out) el esclavo responde. SCK (serial clock) es a lo que se denomina la señal del reloj. [15]

En la tabla 4, se especifica que pines del núcleo están conectados con los pines de la pantalla OLED. [16]

NÚCLEO-L432KC	PANTALLA OLED
GND	GND
+5V	VCC
(D13) SPI1_SCK	D0 (Clock)
(D11) SPI1_MOSI	D1 (Data)
D6	RES (Reset)
(D9) TIM1-CH1	DC (Data/Command)
(D12) SPI1_MISO	CS (Chip select)

Tabla 4

Por la línea MOSI el microcontrolador envía datos a la pantalla, se conecta el pin MOSI del núcleo con el pin Data. Por la línea MISO el esclavo se comunica con el maestro, para ellos se conecta el pin MISO del núcleo con el pin Chip Select. El pin RES y el DC, se conectan con cualquier pin digital del núcleo. [17]

La entrada de USB proporciona 5 V y el Núcleo la baja a 3.3 V para alimentar a los periféricos que funcionan a esa tensión y para alimentar al MCU. El sensor es alimentado por 3.3 V y la pantalla por 5 V.

3.2.3. PULSADOR

En este proyecto se utilizan dos pulsadores. El primer pulsador nos permite desplazarnos por las diferentes opciones que ofrece el menú y el segundo pulsador reinicia el Núcleo-L432KC cuando es pulsado.

Un pulsador consta de 4 patas, conectadas dos a dos. Cuando se presiona el pulsador, se unen los pares de patas, cerrando un circuito que estaba abierto o abriendo uno cerrado. Esto queda mostrado en la Ilustración 10.

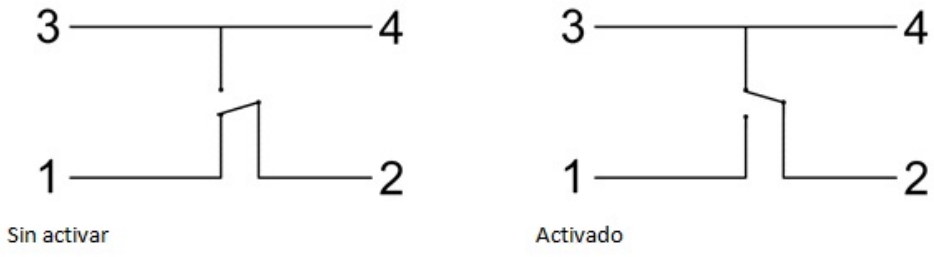

Ilustración 10. Funcionamiento del pulsador

Ilustración 11. Pulsador

Para el conexionado del primer pulsador, se ha utilizado un circuito con resistencia Pull-Up. Para el esquema de Pull-Up, hemos conectado un pin del pulsador a una resistencia y ésta a una fuente de alimentación de 3V. En serie con este primer pin, se une con un cable al núcleo a una entrada analógica. La resistencia utilizada tiene un valor 15 KΩ. La otra pata se conecta a tierra.

Cuando el interruptor está sin pulsar, la corriente circula desde la fuente de alimentación al pin dando un valor lógico de HIGH (1). Pero si el interruptor se pulsa, la corriente va hacia tierra (GND) y el estado pasa a ser LOW (0). [18]

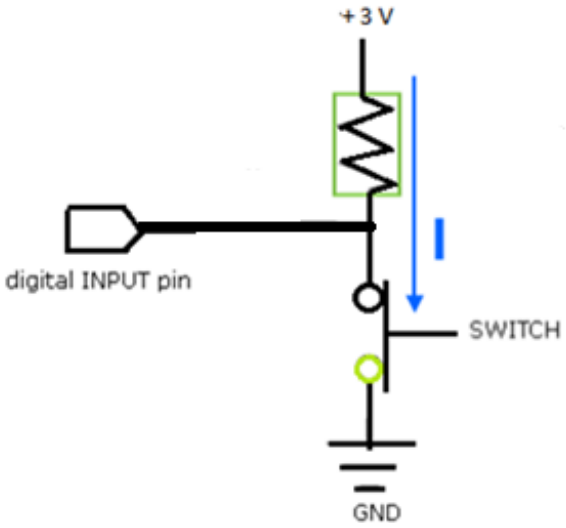

Ilustración 12. Circuito Pull-Up con botón pulsado

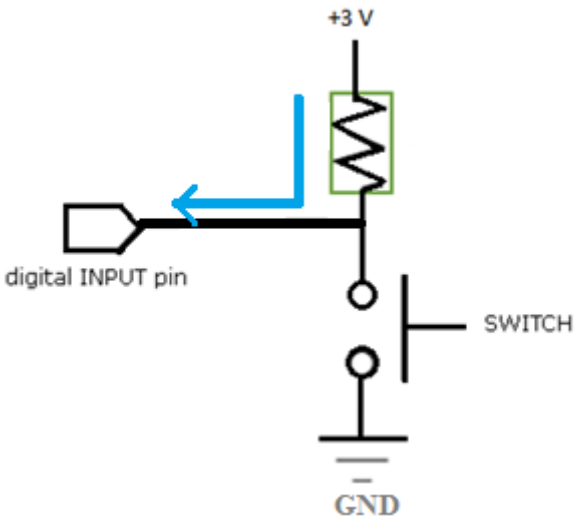

Ilustración 13. Circuito Pull-Up con botón sin pulsar

Con el segundo pulsador, una de las patas se conecta a tierra con un cable y la otra pata se conecta mediante un cable al pin "RST". Cuando el botón no esté presionado, el pin está en su estado normal, pero cuando se pulsa, el pin recibe los 0V de la tensión de tierra y su estado cambia a LOW, y el núcleo se reiniciará. [19]

El controlador de entrada de pin RST utiliza la tecnología CMOS. Está conectado a una resistencia permanente de Pull-Up. Se representa en la Ilustración 14. [8]

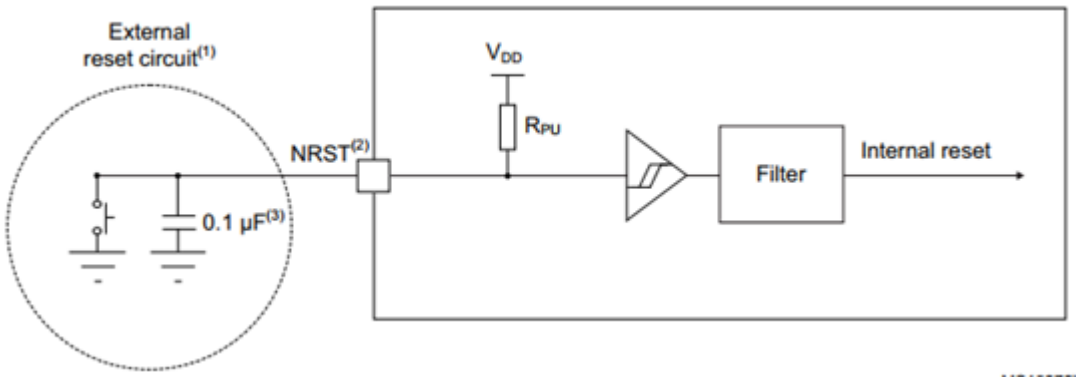

Ilustración 14. Circuito NRST del Núcleo-L432 con protección de clavija NRST recomendada.

3.2.3.1. NÚCLEO CON PULSADOR

En la siguiente tabla se muestran las entradas del núcleo a las que se conectan cada botón mediante un cable.

NUCLEO-L432KC	
PA_ 0	**BOTÓN 1**
RST (CN3)	**BOTÓN 2**

Tabla 5

3.2.4. CONEXIONADO DEL CONJUNTO

A continuación, en la Ilustración 15 se muestra el conexionado de todos los dispositivos. Y en la Ilustración 16, se muestra una fotografía del sistema real, conectados todos los dispositivos entre sí como se describe anteriormente.

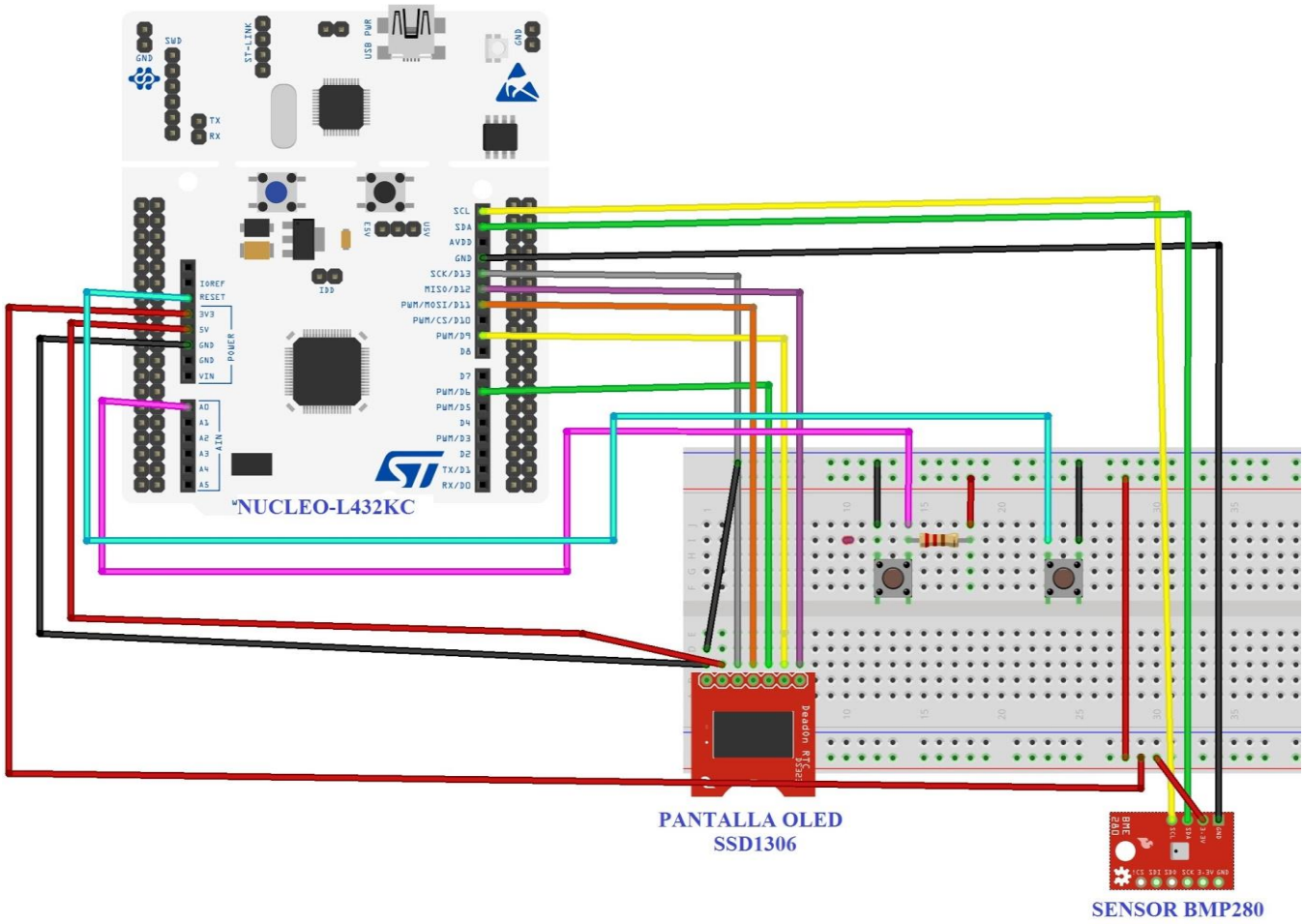

Ilustración 15. Conexionado del conjunto con Fritzing

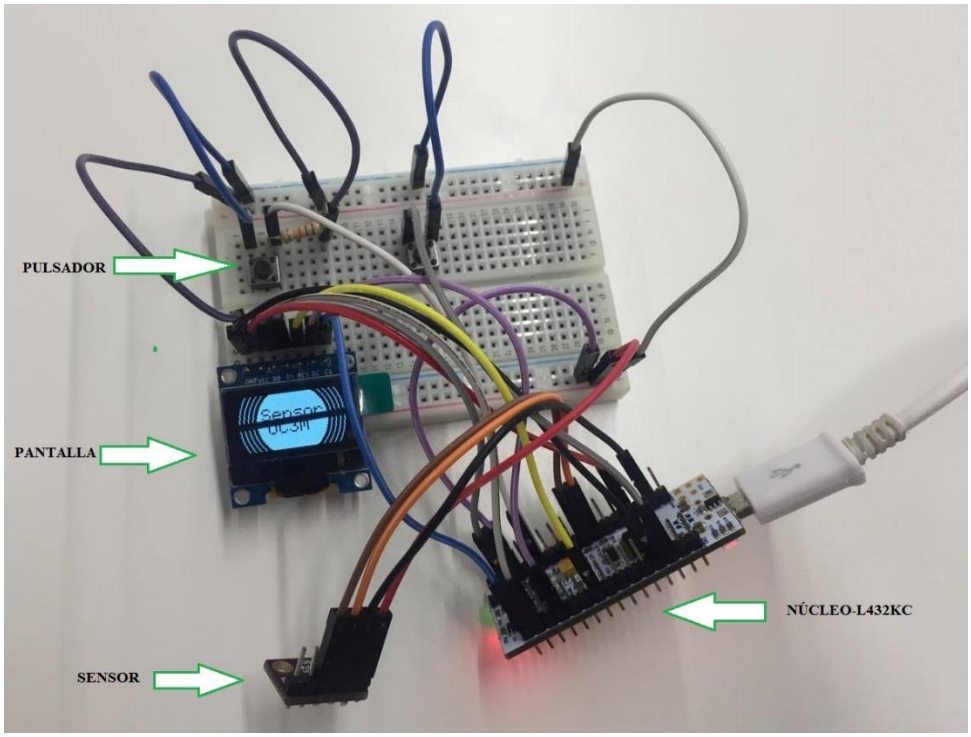

Ilustración 16. Fotografía del conjunto

4. SOFTWARE

4.1. HERRAMIENTAS DE DESARROLLO DE SOFTWARE

Antes de comenzar, el usuario debe seguir los requisitos del sistema.

- Instalar el entorno de desarrollo integrado (IDE) preferido.
- Instalar el controlador ST-LINK / V2-1.
- Descargar el firmware STM32 Núcleo desde la web *www.st.com/stm32nucleo*.
- Conectar el CN1 de la tarjeta Núcleo al puerto USB del PC.

4.1.1. IDEs

Las familias STM32 incluyen IDEs de entorno de desarrollo integrado tradicionales con compiladores C / C++ y depuradores de terceros, completadas con herramientas de STMicroelectronics.

Las cadenas de herramientas que sorportan todas las placas son las siguientes.

• EWARM v7.10.3 o posterior

- Edición de evaluación de 30 días
- Edición QuickStart limitada de 32 kilobytes (limitación de 16 kbytes para Cortex M0)

• MDK-ARM v5.17 o posterior

- MDK-Lite (limitación de tamaño de código de 32-Kbyte)

• TrueSTUDIO Lite v5 o posterior

- Ninguna limitación

• SW4STM32 v1.5 y posterior

- Ninguna limitación [20]

4.1.2. ST-LINK / V2-1

La herramienta de programación y depuración ST-LINK / V2-1 está integrada en la placa STM32 Nucleo-32. El ST-LINK / V2-1 hace que la tarjeta STM32 Nucleo-32 mbed esté habilitada. Esta interfaz necesita un controlador USB para ser instalado. Este controlador está disponible en el sitio web "*www.st.com*".
La versión ST-LINK / V2-1 es una evolución de la versión ST-LINK / V2. Los cambios con respecto a la versión anterior son:

- Reenumeración del software USB.
- Interfaz de almacenamiento masivo en USB.
- Interfaz de puerto COM virtual en USB.
- Solicitud de administración de energía USB para más de 100 mA de encendido USB.

Las funciones no soportadas por esta versión son:

- Interfaz SWIM.
- La tensión de aplicación mínima soportada depende de la implementación del hardware.

ST-LINK / V2- 1 tiene un conector SWD (Serial Wire Debug). SWD es una interfaz eléctrica de 2 pines que reemplaza el puerto JTAG de 5 pines. Utiliza un protocolo de cable bidireccional estándar de la CPU ARM. De esta forma, se permite que el depurador se convierta en otro maestro de bus AMBA para el acceso a la memoria del sistema y a los registros periféricos o de depuración. La velocidad de transmisión de datos es de hasta 4 Mbytes / s a 50 MHz. SWD también tiene una función de detección de errores. En dispositivos JTAG con capacidad SWD, el TMS y el TCK se utilizan como señales SWDIO y SWCLK, proporcionando programadores de modo dual. [21]

SWD es compatible con todos los procesadores ARM y cualquier procesador que utilice JTAG para depuración y proporciona acceso a registros de depuración en procesadores Cortex ™ (A, R, M) y la infraestructura de depuración de CoreSight. [22]

4.1.3. RTC

En la programación de este proyecto, se ha utilizado el reloj de tiempo real para el diseño de la gráfica de la temperatura. Pues gracias al reloj se ha ido registrando el valor de temperatura cada X tiempo.

El núcleo tiene integrado un RTC de baja potencia. Este reloj es un temporizador y/o contador independiente al sistema que recibe energía del suministro VDD. Es capaz de mantenerse activo sin fuentes de alimentación externas. No se reinicia por un sistema o reinicio de energía, o cuando el dispositivo se despierta del modo apagado o espera. [20]

Las fuentes de reloj RTC pueden ser: [23]

- Un cristal externo de 32.768 kHz (LSE)
- Un resonador u oscilador externo (LSE)
- El oscilador RC interno de baja potencia (LSI, con frecuencia típica de 32 kHz)
- El reloj externo de alta velocidad (HSE) dividido por 32.

4.2. PROGRAMA PRINCIPAL

Una vez estén instaladas las herramientas del software y estén conectados todos los periféricos con el núcleo, se diseña la programación en la plataforma “mbed”.

El programa principal, muestra por pantalla las diferentes opciones que tiene el menú. Cada vez que se pulsa el primer botón, se pasa de una opción a otra. La primera opción muestra por pantalla la presión y la altitud registrada en ese mismo instante. La segunda opción muestra por pantalla el dato de la temperatura registrada en ese instante. Por último, en la tercera opción, se dibuja por pantalla una gráfica en la que se representa la temperatura registrada cada 30 minutos. Una vez que se han guardado los 32 datos de temperatura que se representan en la gráfica, en el menú 3 después de la gráfica se muestran los datos de temperatura máximo y mínimo del total.

Además, se ha implementado un segundo botón para ofrecer al usuario la posibilidad de reiniciar los datos de temperatura guardados en la gráfica.

4.2.1. LIBRERÍAS PARA LOS PERIFÉRICOS

Las librerías en informática son una colección de recursos no volátiles utilizados por los programas informáticos para desarrollar un software. Incluyen datos de configuración, datos de ayuda, código y subrutinas prescritas.

- Librerías para el sensor BMP280
  - *BME280.h*
  - *BME280.cpp.*

Para ello, se define en el main, la variable "sensor" de la clase BME280, definida a su vez en la librería. Se define así: *BME280 sensor (I2C_SDA, I2C_SCL)*. Lo que está dentro del paréntesis son la denominación de los pines por los que se mantiene la comunicación, en este caso I2C, entre el microcontrolador y el sensor.

Las funciones que utilizo asociadas a esta clase son, y con las que inicio la comunicación con el sensor, son:

- *Sensor.getTemperature().*

  Al llamar a esta función en el main se registra un dato de temperatura.
- *Sensor.getPressure().*

  Con esta función se registra un dato de presión.

- Librerías para la pantalla SSD1306

Las librerías de este periférico son obtenidas de la comunidad Adafruit, que provee de un código fuente. [24]

- *Adafruit_GFX.cpp*
- *Adafruit_GFX.h*
- *Adafruit_GFX_Config.h*
- *Adafruit_SSD1306.cpp*
- *Adafruit_SSD1306.h*
- *glcdfont.h*

Para la pantalla, se define en el main la variable "display", de la clase Adafruit_SSd1306_SPI. Se define en el main de esta forma: Adafruit_SSD1306_SPI display (SPI_MOSI, SPI_CLK, SPI_CS, DC, RST, 64, 128). Lo que hay dentro del paréntesis son los nombres de los pines por los que se mantiene la comunicación, en este caso SPI, entre el microcontrolador y la pantalla. Las funciones utilizadas para la pantalla vienen definidas en las librerías. Éstas son:

- *void drawFastHLine(int16_t x, int16_t y, int16_t w, uint16_t color)*

  Con esta función se dibuja una recta horizontal, se define el punto X, Y, donde empieza un extremo de la recta. W es la anchura, y color es el color.

- *void drawLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t, color)*

  Se dibuja una recta. X0, Y0 son las coordenadas de un extremo, y X1,Y1 son las coordenadas del otro extremo.

- *void drawFastVLine(int16_t x, int16_t y, int16_t h, uint16_t color)*

  Con esta función se dibuja una recta vertical. Siendo h la altura.

- *void drawRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color)*

  Se dibuja un rectángulo.

- *void fillScreen(uint16_t color)*

  Se pinta toda la pantalla de un color.

- *void drawCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)*
  Se dibuja un circulo, siendo r el radio.
- *void fillCircle(int16_t x0, int16_t y0, int16_t r, uint16_t color)*

  Se pinta el interior circulo de un mismo color.

- *void setTextCursor(int16_t x, int16_t y)*
  Se sitúa un texto en una posición determinada de la pantalla.
- *void setTextSize(uint8_t s)*

  Se establece un tamaño de letra para un texto que se muestra por pantalla.

- *void setTextColor(uint16_t c)*
  Se establece un color para el texto que se muestra por pantalla.

Para escribir un texto con "printf" es necesario definirlo así: *display.printf();* para que el texto se muestre por la pantalla

- Librerías para el pulsador.

Para utilizar un pulsador, es necesario definir en el main una variable, en este caso, "botón", de la clase DigitalIn, definida en la librería de la propia plataforma mbed. Se ha definido de esta forma: DigitalIn Boton (PA_0). PA_0 es una entrada digital del Núcleo-L432kc. De este modo, se crea una entrada digital para comunicarse con el pulsador.

4.2.2. FUNCIONES

La función principal es la función main. La ejecución del programa se inicia con las instrucciones contenidas en su interior. Las funciones secundarias son llamadas desde la función main. El uso de funciones permite un manejo de datos más eficiente y un diseño del programa más estructurado.

En este proyecto se han desarrollado funciones secundarias para el diseño de la gráfica, mostrado en la opción 3 del menú, y para el uso del pulsador.

Para el diseño de la gráfica, teniendo en cuenta las dimensiones de la pantalla, 128 pixeles de ancho y 64 pixeles de largo, se registran un total de 33 datos de temperatura a lo largo del eje X. Inicialmente se dibujan 24 datos, y una vez dibujados estos datos y guardado el dato número 25, el diseño de la gráfica se modifica para poder mostrar un total de 33 datos.

El objetivo inicial es mostrar 24 datos de temperatura a lo largo de 12 horas, registrados y guardados en un vector cada 30 minutos.

Debido a las dimensiones de la pantalla, si el primer dato de temperatura es pintado en X igual a 11.2, quedando restantes 116.8 pixeles, es posible aumentar el número de datos de temperatura dibujados. Se consideró que representar 33 datos en el espacio de 116.8 pixeles de ancho eran suficientes, de este modo se representarían 33 datos de temperatura en un tiempo de 16 horas y media.

A continuación, se explican las funciones desarrolladas.

- *void ReajusteX (int i, int H)*

Esta función dibuja una recta horizontal, el eje X, y dibuja "|" cada vez que hay un dato nuevo de temperatura registrado. Esta función recibe el valor de la variable "i" y "H". La variable "i" indica el número de datos de temperatura que se han registrado. La variable "H" es la distancia a la que tengo que pintar cada dato de temperatura.

Inicialmente, la variable H toma el valor de 5, pero una vez registrados 24 datos de temperatura, la variable H toma el valor de 3.5. De este modo, al reducir el valor de H, se disminuye la distancia que separa cada dato de temperatura que se pinta, consiguiendo dibujar por pantalla un total de 33 datos.

Para dibujar la recta del eje X, se utiliza la siguiente función: display.drawFastHLine().

- *void ReajusteY (float Tmax, float Tmin)*

La función recibe el valor de la variable Tmax, igual al valor máximo de temperatura más 5, y el valor de la variable Tmin, igual al valor mínimo de temperatura registrado menos 5.

Estos dos valores son necesarios para el cálculo de la distancia entre cada dato de temperatura, es decir, el rango del conjunto de datos de temperatura que se muestran. Para el cálculo del rango se ha utilizado esta fórmula:

$$R = \frac{54}{T_{min} - T_{max}}$$

Teniendo un total de 64 pixeles de largo en la pantalla y guardando 10 pixeles para el eje X, quedan disponibles 54 pixeles para representar el eje Y. Esta fórmula me permite conocer el valor del rango dependiendo de los extremos superior e inferior que toma la temperatura.

De esta forma, cada vez que varíe el dato de la temperatura máxima y/o mínima, se vuelve a calcular el valor de la variable R, para cambiar las posiciones de los datos de temperatura en el eje Y.

La primera ejecución de esta función es dibujar una recta vertical para el eje Y. A continuación, se muestran algunos valores de este eje. Se muestra por pantalla el máximo valor de

temperatura, desde ahí se muestran los siguientes valores de temperatura inferiores con una diferencia de 10, es decir, si el máximo valor es el 31, se muestran el 21, el 11… hasta que se alcanza el límite inferior de la recta.

Además, se dibuja un guión "-" cada 5 datos de temperatura, y así facilitar la lectura de la gráfica.

Para dibujar la recta del eje Y, se llama a la siguiente función display.drawFastVLine().

- *void Redibujar (float Tmax, float Tmin, float T[N], int i, int H)*

Esta función dibuja los datos de temperatura registrados, uniendo mediante rectas puntos de temperatura. Recibe los valores de las variables Tmax y Tmin para situar el dato en el eje Y según la propia escala de este eje. El primer dato que se dibuja empezará 11.2 pixeles a la derecha desde el extremo izquierdo de la pantalla, para dejar ese espacio a la función ReajusteY.

Del mismo modo que en la función que "reajusta" el eje Y, en esta función se vuelve a calcular el valor de la variable R. Esto es necesario porque los valores máximos y mínimos de temperatura no son constantes y obligan a redibujar la gráfica para poder mostrar los datos en una misma escala en el eje Y. En el programa principal cada vez que se registra un nuevo dato de temperatura y se guarda en el vector T [], se comprueba si este dato es mayor al límite superior o si es menor al límite inferior establecido.

La función también recibe el vector de temperatura donde se guardan los datos registrados cada 30 minutos, y los valores "i" y "H" para dibujar el dato de temperatura según el eje X.

Se dibuja la recta que une un dato de temperatura nuevo con uno ya registrado anteriormente con la función: display.drawLine().

- *void Fin_grafica (int i, float T[N])*

Esta función muestra por pantalla el valor de la temperatura máxima y el de la mínima de los 33 datos guardados en el vector de temperatura.

Esta función recibe el valor de la variable "i" y el vector de temperatura donde se han guardados

los 33 datos. Con la ayuda de un *if*, compara las temperaturas para asignar la máxima y la mínima. Una vez asignados los valores máximos y mínimos, estos se muestran por pantalla.

Esta función, es llamada en la opción 3 del menú una vez que la variable "i" toma el valor de 33, es decir, ya se han registrados los 33 datos y no se dibuja ningún dato nuevo en la gráfica.

Las siguientes funciones se emplean para el manejo de los pulsadores:

- *int Switch (int boton)*

La función recibe la variable botón. Y devuelve un entero al programa principal.

Esta función se ha desarrollado, de tal forma que registre cambios en el pulsador. Si el pulsador no está pulsado, y en un momento dado se pulsa, se registra un cambio.

Cuando el botón no está pulsado circula corriente y cuando el botón se pulsa, el núcleo deja de recibir esa corriente. El cambio de circulación de corriente es detectado por el núcleo y al presenciarse un cambio, la función devuelve un 1 al programa principal. En caso contrario, si no se registra dicho cambio, la función devuelve un 0.

Además, si el botón es pulsado de forma prolongada, la función no devuelve un 1, pues no se registra un cambio de evento, de no estar pulsado a sí estarlo.

- *int Opcion (int boton, int menu)*

Esta función recibe el valor de la variable botón y menú y devuelve un entero.

Dentro de esta función se llama a la función *int Switch(int boton),* y si ésta devuelve un 1 (porque se ha pulsado el botón y se ha detecta el cambio) se suma 1 al valor del menú. Este nuevo valor del menú es devuelto por la función.

De esta forma, se suma una unidad al valor de la variable menú, para poder pasar a la siguiente opción en el programa principal.

4.2.3. PROGRAMA PRINCIPAL

A continuación, se va explica de forma detallada el contenido del programa principal.

Este programa cuenta con un menú de opciones, desarrollado con la función *Switch case.* Con un primer botón nos desplazamos por las opciones del menú y con un segundo botón, reiniciamos el núcleo. El botón de reinicio, independientemente de lo que se esté mostrando por pantalla, una vez que se pulsa se acciona.

En primer lugar, una vez inicializadas las variables, se inicializa el reloj, y durante todo el programa se comprueban cuántos segundos ha estado en funcionamiento el núcleo. De este modo, mediante un *if*, cada 30 minutos guarda la temperatura ambiental, y se asignan los valores de la temperatura máxima y la temperatura mínima comparándolos con el último dato de temperatura guardado en el vector T []. Si un valor de temperatura está fuera de los límites, el nuevo límite pasa a valer ese dato más 5 unidades si se trata del valor máximo de temperatura, o menos 5 unidades si se trata del límite inferior.

Además, en este *if*, se comprueba si se han guardado 24 datos. Si es así, el valor de la variable "H" cambia a 3.5. Dicho valor es utilizado para dibujar la gráfica de temperatura, es la distancia entre un dato de temperatura y otro.

Durante todo el programa, se comprueba si el primer botón es pulsado para cambiar la opción que se muestra por pantalla.

Cuando el programa entra en una opción del menú, se muestra por pantalla esa opción durante un tiempo inferior a un segundo y siempre se regresa al menú principal para comprobar el *if.* De este modo, cada vez que se sale de un menú, se borra el contenido de la pantalla, llamando a la función: display.clearDisplay().

En la opción 3, si está siendo mostrada por pantalla y se pulsa el botón para cambiar a la opción 1, el valor de la variable menú pasaría a valer 4. Para ello, se comprueba el valor de la variable con un *if*, y se le asigna el valor 1 al menú.

- Opción 0

Es la primera opción que se muestra por pantalla cuando se alimenta al núcleo. Y tras haberse mostrado por pantalla las demás opciones, si el botón de reinicio es pulsado, el programa regresa a esta opción del menú.

En esta opción se inicializan las variables y el reloj. Al ser la opción de inicio del menú, se muestra por pantalla un logotipo compuesto por un círculo de color blanco, rodeado por más círculos y un texto en negro: "SENSOR UC3M".

Además, se registra la temperatura en ese momento, y se asigna el primer valor de la variable Tmax igual a la temperatura más 5, y Tmin igual a la temperatura menos 5.

Una vez que el botón para desplazarse por el menú se pulsa, el reloj empieza a contar los segundos.

- Opción 1

La opción 1 del menú muestra por pantalla la presión atmosférica y la altitud.

Para ello, se llama a la función *sensor.getPressure()*. La función devuelve el valor de la presión atmosférica medido por el sensor BMP280. Este valor se guardar en la variable presión para ser mostrada por pantalla. Para calcular la altitud se utiliza una fórmula que depende directamente del dato de la presión atmosférica y de la presión al nivel de mar.

$$Altitud = 44307.69 * \left[1 - \left(\frac{Presion}{Po}\right)^{0.190295}\right] \text{ (hPa)}$$

Po es la presión al nivel del mar igual a 1013.25 hPa.

Estos dos datos se muestran por la pantalla en el mismo instante. La altitud se muestra en metros y la presión en hPa.

- Opción 2

La opción 2 del menú muestra por pantalla el valor de la temperatura ambiental.

Para ello, en esta opción se llama a la función *sensor.getTemperature()*. Esta función nos devuelve el valor de la temperatura que el sensor BMP280 ha registrado. Este valor no se guarda en ninguna variable. La temperatura se muestra en ºC.

- Opción 3

La opción 3 del menú muestra por pantalla la gráfica de la temperatura con respecto al tiempo, y una vez que se han guardado 33 datos de temperatura, se muestra por pantalla los valores de temperatura máximos y mínimos alcanzados.

Para ello se llama a las funciones *ReajusteY (), ReajusteX ()* y *Redibujar ().* Estas funciones dibujan los ejes de la gráfica y los datos de temperatura. Conforme pasa el tiempo y se guardan nuevos datos de temperatura se van dibujando éstos en la gráfica.

Esta opción contiene un *if*, que comprueba si se han guardado 33 datos de temperatura. En ese caso, las funciones de la gráfica se llaman otra vez para mostrar durante 3 milésimas de segundo más la gráfica con los 33 datos y se llama a una nueva función, *Fin_grafica ().* Pasadas esas 3 milésimas de segundo se borra la gráfica de la pantalla, para mostrar el valor máximo y el valor mínimo de los 33 datos de temperatura.

4.3. DIAGRAMA DE FLUJO

En la ilustración 20, se representa un diagrama de flujo general. El usuario siempre que quiera desplazarse por el menú debe pulsar el botón. Se ilustra como el botón de reinicio puede ser accionado en cualquier momento del programa.

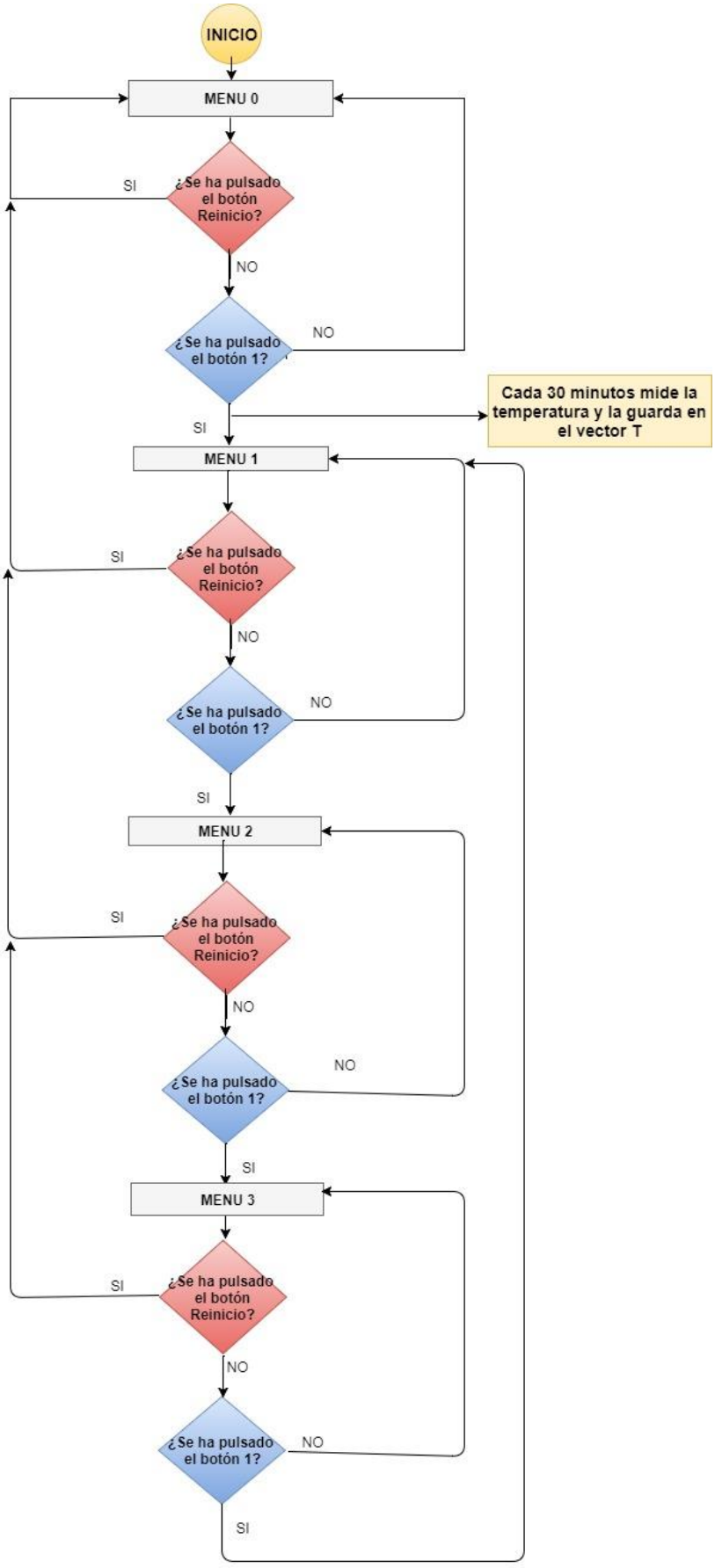

Ilustración 17. Diagrama de Flujo General

5. PRESUPUESTO

A continuación, se muestra el coste de los materiales.

COMPONENTES	**COSTE**
Núcleo-L432KC	9,78 €
Sensor BMP280	2,75 €
Pantalla Oled con micro SSD1306 (128x64)	11,22 €
FUNGIBLES	
Kit de cable conector (20 unidades)	3,00 €
Cable USB Macho	0,99 €
Resistencias (20 unidades)	2,00 €
Micro Pulsador (100 unidades)	2,00 €
Protoboard	3,00 €
TOTAL	**34,74 €**

Tabla 6

La integración del primer prototipo pidiéndolo en China, si se realiza en paquetes de 20 unidades a un precio de 10 €, se estima que costaría 200 €.

Teniendo en cuenta las 250 horas necesarias de trabajo de ingeniería realizados en 5 meses. Si a un ingeniero se le paga un sueldo de 40 € / Hora, un trabajador cobraría un total de 10.000 € por 5 meses de trabajo.

Sumando los distintos costes, se estima un presupuesto total del proyecto de 1.234,74 €.

6. CONCLUSIONES Y TRABAJOS FUTUROS

- Conclusiones

Los objetivos que se definieron se han cumplido de manera satisfactoria, se ha desarrollado un sistema de adquisición de datos. Como resultado del estudio del interconexionado de los elementos del hardware, se ha conectado la placa de desarrollo, con los demás periféricos, sensor BMP280, pantalla OLED y pulsadores.

A raíz de esto se ha desarrollado el software, con la ayuda de la plataforma "mbed", que ha resultado ser muy cómoda y sencilla para la elaboración del software. Además, a partir de las librerías de los periféricos, se ha escrito un código para el funcionamiento del conjunto de los dispositivos. El correcto funcionamiento de este código se muestra por la pantalla Oled.

El microcontrolador y los periféricos seleccionados para este proyecto se han alimentado con tensiones bajas para reducir el consumo de energía. Los materiales utilizados son de bajo coste.

- Futuros trabajos.

Este proyecto admite numerosas ampliaciones. En primer lugar, el conjunto podría integrarse en un sistema más pequeño y dejar de ser un prototipo.

En cuanto al uso de los pulsadores, se pueden utilizar para introducir el valor de la presión actual y poder calibrarla. Incluso, introducir el valor de la altura a la que el usuario está con respecto al nivel del mar, se podrían calcular diferencias de altura. Se podrían emplear dos botones para desplazarse por el menú, en vez de uno. Adicionalmente, podrían calcularse funciones matemáticas, estadísticas como medias o desviaciones típicas.

El núcleo-L432kc se alimenta a través de un cable USB conectado al ordenador. En el futuro a través de baterías, que se conectaran al pin VIN, será posible desarrollarlo como sistema portátil. Adicionalmente, se pueden incorporar fuentes alternativas, como el "Energy Harvesting", para sistemas de captura de energía.

7. BIBLIOGRAFÍA

[1] Historia sobre internet de las cosas. [En línea] [10 de Junio de 2017] Disponible en: http://www.sorayapaniagua.com/2012/04/15/un-poco-de-historia-sobre-internet- de-las-cosas/

[2] El Origen del IoT. [En línea] [10 de Junio de 2017] Disponible en: http://www.bcendon.com/el-origen-del-iot/

[3] Internet de las Cosas. [En línea] [10 de Junio de 2017] Disponible en: http://www.quees.info/que-es-internet-de-las-cosas.html

[4] P. Arora and P. Adodraj, "Portable Game Controller Using STM32,"

[5] ST. life.argumented STM32L series. [En línea] [10 de Junio de 2017] Disponible en: http://www.st.com/content/ccc/resource/sales_and_marketing/promotional_material/brochure/6c/48/c0/f1/bb/35/4a/b4/brstm32ulp.pdf/files/brstm32ulp.pdf/jcr:content/translations/en.brstm32ulp.pdf

[6] LUIS LLAMAS. BUS SPI. [En línea] [10 de Junio de 2017] Disponible en: https://www.luisllamas.es/arduino-spi/

[7] LUIS LLAMAS. BUS I2C. [En línea] [10 de Junio de 2017] Disponible en: https://www.luisllamas.es/arduino-i2c/

[8] ST. life.argumented. Data sheet. [En línea] [10 de Junio de 2017] Disponible en: https://www.google.com/url?q=http://www.st.com/content/ccc/resource/technical/document/datasheet/24/01/9f/59/f0/83/47/fc/DM00257205.pdf/files/DM00257205.pdf/jcr:content/translations/en.DM00257205.pdf&sa=D&ust=1497554556179000&usg=AFQjCNGmKvbpVGuI1boOhca5GeDf-ADgBQ

[9] ST. Life.argumented. UM1956 User manual. [En línea] [10 de Junio de 2017] Disponible en: http://www.st.com/content/ccc/resource/technical/document/user_manual/e3/0e/88/05/e8/74/43/a0/DM00231744.pdf/files/DM00231744.pdf/jcr:content/translations/en.DM00231744.pdf

[10] Arduino ZERO. [En línea] [15 de Junio de 2017] Disponible en: http://www.web-robotica.com/arduino/placas-arduino/arduino-zero-genuino-zero-caracteristicas-tecnicas

[11] BOSCH. [En línea] [[10 de Junio de 2017] Disponible en: https://www.bosch-sensortec.com/bst/products/all_products/bmp280

[12] B. Sensortec, "Digital pressure sensor," May 7th, 2015. http://www.electrodragon.com/w/images/8/8c/BMP280.pdf

[13] SOLOMON SYSTECH SEMICONDUCTOR TECHNICAL DATA. [En línea] [10 de Junio de 2017] Disponible en: https://www.olimex.com/Products/Modules/LCD/MOD-OLED-128x64/resources/SSD1306.pdf

[14] ARMBED: NUCLEO-L432KC. [En línea] [10 de Junio de 2017] Disponible en: https://developer.mbed.org/platforms/ST-Nucleo-L432KC/

[15] PROMETEC. BUS SPI. [En línea] [10 de Junio de 2017] Disponible en: http://www.prometec.net/bus-spi/

[16] Gráficos y Oled 0.96” 128x64. [En línea] [10 de Junio de 2017] Disponible en: http://www.prometec.net/oled-graf/

[17] ADAFRUIT. [En línea] [10 de Junio de 2017] Disponible en: https://learn.adafruit.com/ssd1306-oled-displays-with-raspberry-pi-and-beaglebone-black/wiring

[18] R. F. J. Stopel and J. Tjia, "No title," Pull-Up Circuit, 2009.

[19] Y. Kawasaki, S. Hashimoto and K. Yamamoto, "No title," System for Recognizing of a Device Connection State by Reading Structure Information Data which Produced by Pull-Up Resistor and Pull-Down Resistor, 2003.

[20] ST. Life.argumented UM1727 User manual. [En línea] [10 de Junio de 2017]Disponible en: http://www.st.com/content/ccc/resource/technical/document/user_manual/1b/03/1b/b4/88/20/4e/cd/DM00105928.pdf/files/DM00105928.pdf/jcr:content/translations/en.DM00105928.pdf

[21] ARMBED. Serial Wire Debug [En línea] [10 de Junio de 2017] Disponible en: https://www.arm.com/products/processors/serial-wire-debug.php

[22] TN1235 Technical note. [En línea] [10 de Junio de 2017] Disponible en: http://www.st.com/content/ccc/resource/technical/document/technical_note/group0/30/c8/1d/0f/15/62/46/ef/DM00290229/files/DM00290229.pdf/jcr:content/translations/en.DM00290229.pdf

[23] ST. Life.argumented. RTC [En línea] [10 de Junio de 2017] Disponible en: http://www.st.com/en/clocks-and-timers/real-time-clock-rtc-ics.html

[24] Adafruit. [En línea] [10 de Junio de 2017] Disponible en: https://github.com/adafruit/Adafruit_SSD1306

8. ANEXOS

8.1. CÓDIGO

A continuación, se muestran las líneas de código que han sido elaboradas en la plataforma "mbed".

```cpp
#include "mbed.h"
#include <stdlib.h>
#define PRESSED 0
#define NOT_PRESSED 1
//PANTALLA
#include "Adafruit_SSD1306.h"
 #define SPI_CLK     PB_3
 #define SPI_MOSI    PB_5
 #define SPI_MISO    NC
 #define SPI_CS      PB_4
 #define DC          PA_8
 #define RST         PB_1
//SENSOR
#include "device.h"
#include<math.h>
#include "BME280.h"
#include "stdio.h"
#define N 300
Serial pc(USBTX, USBRX);
#if defined(TARGET_LPC1768)
BME280 sensor(p28, p27);
#else
BME280 sensor(I2C_SDA, I2C_SCL);
#endif
Adafruit_SSD1306_SPI display(SPI_MOSI, SPI_CLK, SPI_CS, DC, RST, 64, 128);

//FUNCIONES
int Switch(int boton);
int Opcion(int boton, int menu);
void Redibujar (float Tmax, float Tmin, float T[N], int i, int H);
void ReajusteX (int i, int H);
void ReajusteY (float Tmax, float Tmin);
void Fin_grafica (int i, float T[N]);
DigitalIn Boton (PA_0); //A0

int main()
{
    display.clearDisplay();
    Boton.mode(PullUp);
    double altitud;
    double presion;
    int H=5; //PARA COLOCAR EN EJE X LAS BARRAS |
    float T[N]; //GUARDO TEMPERATURA
    float Tmax, Tmin;
    int i=0; //CONTADOR.z LO UTILIZO EN EL VECTOR T[N]
    int menu=0;
    int op=0;
    int sec=0;
    float v;
    time_t seconds;

    while(1){

        seconds = time(NULL);
        sec= seconds;
        menu= Opcion(Boton,op);
        wait(0.1);

        if( (-1800*i) + sec == 1800 ){     //30 mins
        i++;
        v= sensor.getTemperature();
        T[i-1]= v;

            if(i>23){ //H=5 con 24 datos , H=3.5 con mas de 24. REAJUSTE X
            H= 3.5;
            }
            if( v > Tmax){    //Compruebo si mi temperatura es mayor que la maxima
            Tmax=v+5;
            }
            if(v < Tmin){     //Compruebo si mi temperara es menor que la mínima
            Tmin= v-5;
            }
        }

    //MENU ENTRE 1 Y 3
```

```
    if(menu>=4){
    menu=1;
    }

    if( menu != op){  //se pulsa
    op=menu;
    display.clearDisplay();
    }

    switch (menu)
      {
        case 0:  //Menu inicio


       display.drawCircle(64,32,45, WHITE);
       display.drawCircle(64,32,50, WHITE);
       display.drawCircle(64,32,55, WHITE);
       display.drawCircle(64,32,60, WHITE);
       display.fillCircle(64, 32, 40, WHITE);
       display.setTextCursor(6,16);
       display.setTextSize(2);
       display.setTextColor(BLACK);
       display.printf("  Sensor \n\r   UC3M");
       display.display();
       set_time(0);
       Tmax = sensor.getTemperature()+5 ;
       Tmin = sensor.getTemperature()-5 ;
       menu= Opcion(Boton,op);
       wait(0.2);
       break;

       case 1:

       presion = sensor.getPressure();
       altitud = 44307.69* (1 - pow(((double) (presion/1013.25)), 0.190295)); //Calcula la altitud
       display.setTextCursor(2,15);
       display.setTextSize(1);
       display.setTextColor(WHITE);
       display.printf("PRESION = %04.2f hPa\n\r", presion);
       display.printf("\n ALTITUD = %04.2f m \n\r", altitud);
       display.display();
       wait(0.2);
       display.clearDisplay();;
       break;

       case 2:

       display.setTextCursor(5,20);
       display.setTextSize(2);
       display.setTextColor(WHITE);
       display.printf("%2.1f degC", sensor.getTemperature()); //Muestra la temperatura y la presión
       display.display();
       wait(0.2);
       display.clearDisplay();
       break;

       case 3:

       ReajusteY(Tmax, Tmin);
       ReajusteX(i-1, H);
       Redibujar (Tmax, Tmin,  T,  i-1, H );
       display.display();
       wait(0.3);
       display.clearDisplay();

           if(i >33){
           ReajusteY(Tmax, Tmin);
           ReajusteX(i-1, H);
           Redibujar (Tmax, Tmin,  T,  i-1, H );
           display.display();
           wait(0.3);
           display.clearDisplay();
           Fin_grafica (i-1 , T);
           display.display();
           wait(0.4);
           wait(0.2);
           display.clearDisplay();
           }
```

```
153
154           break;
155         }
156       }
157   }

158   void Fin_grafica (int i, float T[N])
159   {
160
161       float TMAX, TMIN;
162       int j;
163
164       TMAX=T[1];
165       TMIN=T[1];
166
167               for(j=0; j<33; j++){
168                   if( T[j]>= TMAX){
169                   TMAX= T[j];
170                   }
171                   if (T[j] <= TMIN){
172                   TMIN = T[j];
173                   }
174               }
175
176               display.setTextCursor(2,15);
177               display.setTextSize(1);
178               display.setTextColor(WHITE);
179               display.printf("T MAX = %2.1f degC ", TMAX);
180               display.printf("\n\nT MIN = %2.1f degC ", TMIN);
181    }
182
183   int Switch(int boton)//RECIBE ESTADO ANTERIOR y ENTRADA DIGITAL
184   {
185       int pulsado;
186       int switchEvent=0 ;
187
188       if(pulsado == NOT_PRESSED){
189           if(!boton){ //presiono y no hay corriente
190               pulsado = PRESSED;
191               switchEvent = 1;
192           }
193       }else{
194           switchEvent=0;
195           if(boton){ //no presiono y Si hay corriente
196               pulsado = NOT_PRESSED;   // Change the button's state to NOT_PRESSED
197           }
198       }
199   return (switchEvent);
200   }
201
202   int Opcion(int boton,int menu)
203   {
204
205       int cambio=0;
206
207       //BOTON DOWN
208        cambio = Switch (boton); //funcion me devuelve 1 si hay cambio/ 0 si no lo hay
209           if(cambio==1){
210               menu +=1;
211           }
212
213   return(menu);
214   }
215
216   void ReajusteY (float Tmax, float Tmin)   //DIBUJA EJE Y ,AJUSTA LA ESCALA DEPENDIENDO DE LAS T max y min
217   {
218       int V=0, cont=0;
219       float R=0;
220       display.drawFastHLine(11.2,60,127, WHITE);
221       display.drawFastVLine(11.2,0, 63, WHITE); //eje y
222       R=(54/(Tmax-Tmin)); //Rango
223
224       while((R*V)<=55){ //mientras este dentro de las dimensiones de la pantalla
225
226           cont++;
227
228           if( cont % 10 == 0){ //cuando cont valga 10,20.. pongo en el eje y el valor: Tmax - 10,20...
229           display.setTextCursor(0,(R*V)+2); //pongo mas 4 para centrar el numero, el numero ocupa 8pixeles verticales
```

```
230         display.setTextSize(1);
231         display.setTextColor(WHITE);
232         display.printf("%2.0f", Tmax-cont);
233         display.setTextCursor(0,0);
234         display.setTextSize(1);
235         display.setTextColor(WHITE);
236         display.printf("%2.0f",Tmax);
237         }
238         if( V %5==0){ //cada 5 datos pongo -
239         display.setTextCursor(0, R*V );
240         display.setTextSize(1);
241         display.setTextColor(WHITE);
242         display.printf(" -");
243         }
244         V++;
245     }
246 }
247
248 void ReajusteX (int i, int H)   //DIBUJA EJE X Y UN "|" CADA VEZ QUE SE PINTA UN DATO EN LA GRAFICA
249 {
250     int j;
251
252     for(j=0; j<=i; j++){
253     display.drawFastHLine(11.2,60,127, WHITE);
254     display.setTextCursor(10.2+(H*j),60);  //maximo 33 datos
255     display.setTextSize(1);
256     display.setTextColor(WHITE);
257     display.printf("|");
258     }
259 }
260
261 void Redibujar (float Tmax, float Tmin, float T[N], int i, int H)
262 {
263     int j;
264     float R;
265
266     R=(54/(Tmax-Tmin)); //Rango
267     for(j=0; j<=i; j++){
268     display.drawLine(11.2+(H*(j)) ,R*(Tmax-T[j-1]) ,11.2+(H*(j+1)),R*(Tmax-T[j]), WHITE);
269     }
270 }
```